基于 3 层 C/S 结构的 MIS 权限设计

厦门大学机电工程系(361005) 彭永谦 柳 旭

摘 要：分析了 3 层 C/S 结构的机理及其优点。提出了一个基于 3 层 C/S 结构的 MIS 权限设计方案,说明了用 VB6 开发 3 层应用系统的权限控制机制的基本技术和方法。

关键词：3 层 C/S MIS 系统 ADO MTS 平台

1 2 层 C/S 结构及其局限性

在传统的 C/S 应用中，客户端实现了用户接口的功能,同时封装了部分或全部应用逻辑。其结构如图 1 所示。

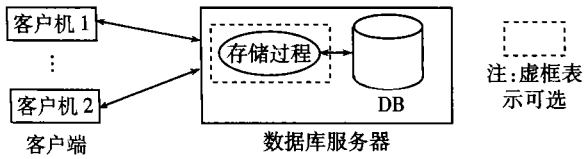

图 1 2 层应用模型

其基本工作方式是客户程序向数据服务器发送 SQL 请求，服务器返回数据和结果，图中虚框中表示存储过程,它与 DBMS 相关,在安全性方面有所提高,但在可伸缩性和可重用性方面仍然不足。在规模较大的应用系统中其局限性表现在以下方面:

(1)效率低下。客户机通过网络连接访问远端数据,不仅降低了本机的性能，而且服务器必须保持同每个活动的客户机连接,也降低了服务器的性能,每个活动的连接占用一部分网络带宽,使网络通信繁忙。同时,应用逻辑驻留在客户机,逻辑组件利用率低。

(2)安全性差。客户端应用程序直接和数据库打交道,因此,客户端拥有对数据库操作的足够权限,致使非法用户能够操作甚至破坏数据库。

(3)维护困难。由于应用逻辑部分或全部封装在客户端,因而不能对这些规则进行集中控制和管理。一旦应用逻辑发生变化,就必须更新客户端程序。当系统规模很大时,维护代价较高。

2 3 层 C/S 结构及优点

随着 INTERNET/INTRANET 应用需求的增长和应用可伸缩性要求的提高,传统的 Client/Server2 层应用模式已不能很好地满足需要,MIS 开发逐步向 3 层以至多层应用发展。

典型的数据库应用开发可分为 3 部分:表示层、应用逻辑层(或称商业逻辑层)、数据服务层。表示层实现用户接口的功能，应用逻辑层根据应用规则实现计算和控制程序的流程等,数据服务层提供数据的访问和维护。

3 层 C/S 结构将应用的 3 层明确地进行分割，使其在逻辑上各自独立,并且单独加以实现,分为客户、应用服务器、数据服务器。与 2 层 C/S 结构相比,其应用逻辑层被明确地划分出来,其结构如图 2 所示。

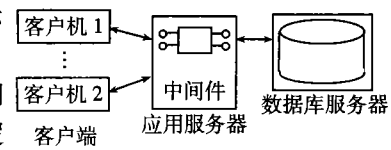

图 2 3 层应用模型

客户机实现表示层，负责与用户交互，并把用户请求通过调用中间层组件(应用逻辑依据面向对象的方法封装在组件中)传递给应用逻辑层,应用逻辑层执行具体的事务逻辑，并通过 SQL 方式向数据服务层提出数据或其它资源的请求。在具体的硬件实现上,应用服务器和数据服务器可位于同一主机上,也可位于不同的主机上,3 层是指逻辑上的划分。

这种划分与传统的 C/S 结构相比,有以下优点:

(1)提高效率。在 3 层 C/S 结构中,客户端应用程序和应用服务器之间实际上只是一些简单的通信协议，这减轻了客户端的负担,也降低了数据服务器的连接代价。另外,应用程序组件可共享与数据库的连接,降低了数据服务器的负担。

(2)安全性加强。安全管理基于组件来授权给客户,客户机不再直接访问数据库,提高了系统数据的安全性。

(3)易于维护。客户端只处理用户接口,不涉及数据的存取;而应用逻辑的更新只对中间层进行修改,不用更新整个系统。

(4)组件可共享和重用。应用逻辑可开发成可重用的二进制组件,供其它应用程序共享和重用,且可镜像到多台服务器上同时运行。

3 3 层 C/S 结构的应用系统的权限设计

2 层 C/S 结构的 MIS 系统，其权限设计实质上也是基于 2 层的。因此,同样具有 2 层 C/S 应用系统的缺陷。众所周知,权限设计是 MIS 的关键部分,其可靠性、安全

性及稳定性直接影响到 MIS 的运行。采用 3 层 C/S 结构把关键业务规则(即应用逻辑)从表示层中分离出来,提高了安全性。同时采用组件来授权给用户,也提高了安全性,另外组件也提高了系统的效率。本文所阐述的方案是采用支持 3 层应用开发的工具 Visual Basic 6.0 企业版开发中间层(组件)和表示层,应用 Windows NT Server 4.0(sp3),MS SQL Server 6.5(sp5a)作为数据库服务器。采用 Visual Basic 6.0 是因为其具有强大的组件开发能力,同时使用 Microsoft 的 Universal Data Access (通用数据接口,包括 ADO 和 OLEDB)能够为不同的数据源提供一致的访问。

本文的权限设计方案是基于对 Windows NT、MS SQL Server 以及一些管理信息系统(MIS)的安全设计的考察,在吸收其优点的基础上提出的,详细说明如下。

3.1 应用系统的分割

3.1.1 数据服务层

虽然直接访问数据库要经过 NT 和 SQL 登录,但这仅仅是必需的,而用户界面上对象的权限无法得以控制。因此,在 SQL 数据库服务器中建立 2 个用于权限管理的表。1 个用户权限表(表名为 UsersRight),包括用户名、用户密码、权限 3 个字段,用户名为主关键字,用户密码通过组件加密,用密文存储,这样即使能通过 SQL Server 或其它方法查看到该表,也无法得到实际的密码,使用密文密码无法登录到 MIS。另外建立 1 个存储过程 SetObjRgtInDb(@username varchar(10),@right varchar(8)),其功能是通过用户权限表中的用户记录来设置该用户数据库对象权限。1 个对象操作权限表 (表名为 PermOperateObj),包括权限、许可对象 2 个字段,权限和许可对象为主关键字。表中存放着关键数据,对其直接操作,存在不安全因素,因此,建立 2 个存储过程 SearchByRight(@right varchar(8)),SearchByObj(@object varchar(20))。通过这 2 张表,可以方便地添加和删除用户,更改用户的密码和权限,进行权限定义,这些操作只需在相应的表中添加和删除记录即可。

3.1.2 应用逻辑层

Microsoft 提供了很多中间件平台和开发工具,MTS (Microsoft Transaction Server) 就是相应的平台,Visual Basic 6.0 就是强大的组件开发工具。本文是通过建立 1 个基于 MTS 的 In-process 组件来实现应用逻辑层。因为用户前端同中间层及中间层与后台数据服务的交互,必将涉及到进程间的通信、分布式计算等问题。而这些问题的解决,可将应用逻辑依据面向对象的方法封装在组件中,相应的组件被部署在中间层。安装在 MTS 上的软件包(包含应用逻辑层组件)可以通过 IIS(Internet Information Server)或 DCOM(分布式组件对象模型)与客户端交互,因此这些需要远程调用的组件的进程间的通信、分布式计算等问题完全由 MTS 完成。

本文中是使用 VB6 开发了 LoginManager.dll 来实现权限管理。该组件针对于客户端的需求采用以下的类及其方法。

(1)类 CheckedIn,有一方法 CheckedIn:该方法用来检查用户登录,声明为 Public Function CheckedIn(ByVal UserName As String,ByVal UserPwd As String) As Long;另外有一友元函数 Decode,用于将表 UsersRight 的 UsersPwd 字段解密,以便与 CheckedIn 函数的 UserPwd 参数进行比较,声明为 Friend Function Decode (ByVal source As String)As String。

(2)类 NameToPermi,有一方法 CheckedObj:该方法通过用户名来获取用户层界面上的对象的操作许可权,声明为 Public Function CheckedObj (ByVal Name As String,ByVal ObjName As String)As Long。通过 Name 在表 UsersRight 中查询权限字段,检查该字段的值是否在存储过程 SearchByObj(@object varchar(20))所检索的结果集中(存储过程使用 ObjName 参数),在结果集中说明有操作该对象的权限。

(3)类 DefineRight,有一方法 DefineRight:该方法用来定义权限,声明为 Public Function DefinedRight(ByVal Name As String,ByVal Right As String,ByVal ObjPermiStr As String) As String。一方面在对象操作权限表(表 PermiOpearateObj)中写入记录;另一方面调用存储过程 SetObjRgtInDb (@username varchar (10), @right varchar (8)) 设置数据库对象的权限。其中 ObjPermiStr 采用格式“#obj1#boj2#…#obji#…#objn”,obji 是写入许可对象字段的值。

(4)另外有用于更改密码、添加用户、删除用户、更改或设置权限的类,在此不做详细的讨论。

3.1.3 表示层

表示层主要反映客户端的需求,包括 3 部分:用户登录系统、运行控制机制、系统维护。具体说明如下。

(1)用户登录系统:检查合法用户和密码,调用 Checked In()函数,同时记下合法用户名。

(2)运行控制机制:采用触发检查方式,即用户一旦选择某项操作即立即激活权限检查,决定其是否有权执行该项操作,否则系统将拒绝执行,调用 CheckedObj()函数。

(3)系统维护:包括更改密码、添加用户、删除用户、设置或更改用户权限、权限定义等 5 部分。对一般用户只可更改密码,而系统管理员则具有所有的权限。

3.2 应用系统 3 层权限设计的实现

3.2.1 数据服务层

(1)建立表 UsersRight、PermiOpreateObj

可通过 SQL Server 的 Manager Tables 窗口建立。

(2)建立存储过程

存储过程 SearchByObj:

```
CREATE PROCEDURE SearchByObj
(@oprobj varchar(20))
AS
    select UsersRgt from PermiOpreateObj where
                                PermiObj=@oprobj
return 0
```

存储过程 SearchByRight:

```
CREATE PROCEDURE SearchByRight
(@right varchar(8))
AS
   select PermiObj from PermiOpreateObj where
                                UsersRgt=@right
return 0
```

存储过程 SetObjRgtInDb (@username varchar(10), @right varchar(8)):由于涉及到具体的数据库,因此不具有一般性。这里给出其实现方法:

(1)可将其加入具有对应权限的组(在 SQL Server 数据库中具有相同数据库对象权限的用户的集合);使用系统存储过程 sp_addgroup。

(2)可以编写对数据库对象授权的代码。

(3)将 SQL Server 设为整体安全模式,即所有用户均使用 NT Server 登录用户名和密码,登录 SQL Server 不需输入用户名和密码。也就是采用信任连接。

3.2.2 应用逻辑层

采用 VB6 开发中间件 LoginManager.dll:

类 CheckedIn:

```
//采用信任连接,假设数据库是 employee,服务器为
                                        //NTSRV
Private Const CON_STR_CONNECTION =" Provider =
SQLOLEDB.1;Integrated Security =SSPI;Persist Security
Info=False;Initial Catalog=employee;Data Source=NTSRV"
Public Function CheckedIn (ByVal UserName As
String,ByVal UserPwd As String)As Long
    On Err GoTo ErrHandler
    Dim adoConn As New ADODB.Connection
    Dim adoCmd As New ADODB.Command
    Dim adoRs As New ADODB.Recordset
    Dim flag As Long
    flag=0     //登录标志。1=成功,0=失败。默认未登录。
    //实例 MTS Automation 库提供的 ObjectContext 对象
    Dim objContext As MTxAS.ObjectContext
    Set objContext=GetObjectContext()
    adoConn.Open(CON_STR_CONNECTION)
    adoCmd.ActiveConnection=adoConn
    adoCmd.CommandType=adCmdTable
    adoCmd.CommandText="UsersRight"
    Set adoRs=adoCmd.Execute
    If adoRs.EOF And adoRs.BOF Then
        GoTo ErrHandler
    End If
    adoRs.MoveFirst
    Do While Not adoRs.EOF
        If UserName=adoRs! UsersName And UserPwd=
                                Decode(adoRs! UserPwd)
        Then
            flag=1
            Exit Do
        End If
        adoRs.MoveNext
    Loop
    If Not objContext Is Nothing Then
        objContext.SetComplete
    End If
        CheckedIn=flag
        Exit Function
    ErrHandler:
        If Not objContext Is Nothing Then
          objContext.SetAbort
        End If
        CheckedIn=flag
        Err.Raise Number:=Err,source:="CheckedIn",
                        Description:=Err.Description
    End Function
```

关于友元函数 Decode( ),可使用任何一种加密、解密算法。

其它有关的类在此不做说明。对以上例可以总结 MTS 编程如下:

在 VB6 中必需引用 Microsoft ActiveX Data Objects 2.0 Library 库和 Microsoft Transaction Server Type Library 库。

MTS 编程模式如下:

```
On Err GoTo ErrHandler
Dim objContext As MTxAS.ObjectContext
Set objContext=GetObjectContext()
//编写正确处理代码。
If Not objContext Is Nothing Then
   objContext.SetComplete //提交事务。
End If
   //CheckedObj=正确的返回值(Sub( )不需要此语
                        //句,只用于 Function( ))
Exit Function
ErrHandler:
   //CheckedObj=错误的返回值(同上)
   If Not objContext Is Nothing Then
   objContext.SetAbort     '回滚事务。
```

```
    End If
    Err.Raise Number:=Err,source:="错误提示！",Description:=Err.Description
End Function
```

使用完 Connection 对象后，并未显示调用 Close 方法释放连接。这是因为 MTS 为中间层组件实现了资源共享池(Resource Pool),所以,与数据库的连接应该由 MTS 协助维护和分配。

3.2.3 表示层

(1)登录系统:调用 CheckedIn( ),记下合法的用户名,如为 CurUser。

(2)运行控制机制:①用户名:CurUser。②用户界面上的操作对象采用所在窗体的名称 _ 对象名，这样就使操作对象名唯一。如一对象的 Name 属性为 cmdMaoYi,操作对象名=cmdMaoYi.Parent.Name& "_" & cmdMaoYi。

③将权限检查放入功能调用模块的首部，由其决定是否执行该操作。例如贸易功能模块可加入代码:

```
If Not NameToPermi(CurUser,cmdMaoYi.Parent.Name_& "_" & cmdMaoYi) Then
    Exit Sub
End If
//贸易功能模块执行代码
```

(3)系统维护:仅说明权限定义。用户界面上给出所有的功能模块,通过复选框选择该权限所包含的功能模块,生成类 DefineRight 中说明的字符串的形式，调用 DefineRight( )函数。

本文主要给出基于 3 层 C/S 结构的应用系统中的权限设计方案,给出的 VB6 代码具有普遍性,不必修改即可应用于 SQL Server 7.0 数据库系统。

(收稿日期:1999-09-01)

(上接第 7 页)

换只是简单地将 NT Server2 上需保护应用程序(APPB)加到 NT Server1 的运行负载上去。

(3)N-Way (N=3,N=4……N=16)

如图 4 所示,N-Way 配置是激活/激活或激活/备用的 1 个有 3~16 个服务器的扩展。服务器 A 被配置为服务器 B 和服务器 C 的备份。而且，服务器 A 可以被配置为除了服务器 B 和服务器 C 的其它服务器的备份。当任何一个服务器发生故障时，被保护的应用程序被从该服务器上转到备用的服务器上。

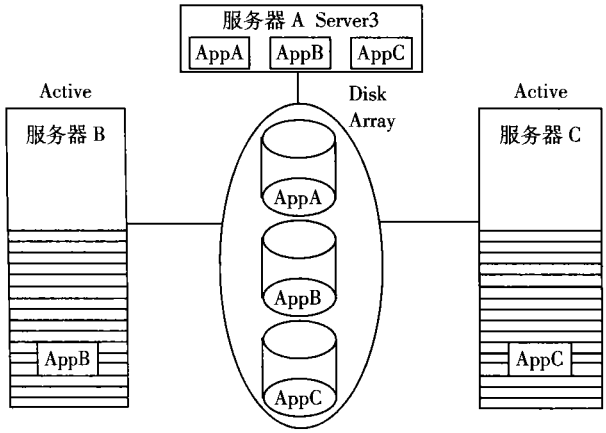

图 4 N-Way 工作方式示意图

在 N-Way 配置中,可以配置 Cascading Recovery(层叠恢复)。当主节点发生故障时，层叠恢复允许多个从属节点被按照一定的优先级次序恢复一个资源或层次。

3 NCR LifeKeeper 的关键特点

1.不用增加任何额外硬件投资,纯软件方式实现双机容错,且对备份机无硬件配置要求。

2.可支持 Notes、Exchange、SQL Server、Sybase、Informix、Oracle、SAP 等多种系统的应用层热恢复。

3.是同时支持 NT 操作系统和 UNIX 平台的容错软件。支持远程灾难备份。

4.支持共享磁盘阵列柜和扩展镜像二种方式,给用户提供了选择上的灵活性,同时也能适应各种机型、网络结构、软件平台及应用系统。

5.LifeKeeper 在扩展镜像或共享磁盘阵列任意方式下,均能实现二台 NT 服务器各自运行不同应用且相互热备份,即实现双 Active 运转模式。

6.使用共享磁盘阵列柜方式时,最多可以支持 16 个节点,远远大于其它类似系统所支持的 2 个节点数。

7.最大限度地保护用户端的应用连续性。用户的硬件资源(如网卡)及软件资源(如 NT 操作系统、数据库管理系统、数据库应用系统、电子邮件系统等) 均能处于 LifeKeeper 的保护之下,当这些被保护资源出现技术故障时,LifeKeeper 可随时实施系统资源切换。如此,LifeKeeper 真正实现了用户硬件或是软件资源发生故障时系统及应用层上的在线热切换。

8.LifeKeeper 占用系统资源极少，不增加网络负荷,且不打扰任何具体应用系统的任何操作。

9.LifeKeeper 真正实现无人值守,全自动地实现应用资源切换,且图形界面操作,简单方便。

10.自投入使用以来,已经历了大量交易高峰的实际考验,其执行效率很高且运行十分稳定可靠。

(收稿日期:1999-12-25)